Estructuras de Datos y Algoritmos

Grupos C y F

Examen segundo cuatrimestre, 31 de mayo de 2019

Normas de realización del examen

1. Escribe en un comentario al principio de cada fichero tu nombre, apellidos, grupo (C o F) y usuario que has recibido para realizar este examen.
2. Debes desarrollar e implementar soluciones para cada uno de los ejercicios, probarlas y entregarlas en el juez automático accesible en la dirección `http://exacrc`.
3. Del enlace **Material para descargar** dentro del juez puedes descargar un archivo comprimido que contiene material que puedes utilizar para la realización del examen (transparencias de clase, implementación de las estructuras de datos, una plantilla de código fuente y ficheros de texto con los casos de prueba de cada ejercicio del enunciado).
4. En el juez te identificarás con el nombre de usuario y contraseña que has recibido al comienzo del examen. El nombre de usuario y contraseña que has estado utilizando durante la evaluación continua **no** son válidos.
5. Escribe comentarios que expliquen tu solución y ayuden a entenderla. Calcula la complejidad de todas las funciones que implementes.
6. Los ficheros con las implementaciones de las estructuras de datos están instalados en el juez, por lo que no es necesario subirlos como parte de tu solución (y conviene no hacerlo).
7. Los ejercicios están identificados con el nombre del tema de la asignatura en el que habrían aparecido si hubieran sido propuestos como ejercicios durante el curso. Para obtener la máxima puntuación, las soluciones deberán seguir los criterios exigidos a los ejercicios de ese tema durante el curso.
8. Tus soluciones serán evaluadas por la profesora independientemente del veredicto del juez automático. Para ello, la profesora tendrá en cuenta **exclusivamente** el último envío que hayas realizado de cada ejercicio.
9. Al terminar el examen, dirígete al puesto del profesor y rellena con tus datos la hoja de firmas. Muéstrale tu documento de identificación.
10. Si te es más comodo, puedes entregar las explicaciones de las soluciones por escrito, en papel.

Ejercicio 1. Tipos de datos lineales (2,5 puntos)

Dado el TAD **queue** define una nueva operación que elimine de la lista enlazada todo elemento que sea múltiplo de su correspondiente anterior en la lista. Ten en cuenta que si un número x_n es múltiplo de su anterior x_{n-1} y este se elimina, entonces x_{n+1} se compara con x_{n-1}, pues x_n ya no existe.

Para resolver este ejercicio se debe extender la clase **queue** del fichero `queue_eda.h`, para ello utiliza la plantilla que se proporciona en el fichero `material/Ej1/mainEjer1.cpp`. Lo importante para el ejercicio es que internamente la cola **queue** está representada con una lista enlazada simple de nodos dinámicos.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso se muestra en dos líneas, la primera contiene el número de elementos de la lista (un número mayor que 0). En la segunda se muestran los elementos de la lista. Termina con `0`.

Entrada de ejemplo

```
8
2 4 2 5 6 9 4 10
8
2 2 2 2 2 2 2 2
5
2 4 3 6 9
5
4 3 6 12 5
6
1 2 3 4 5 6
0
```

Salida

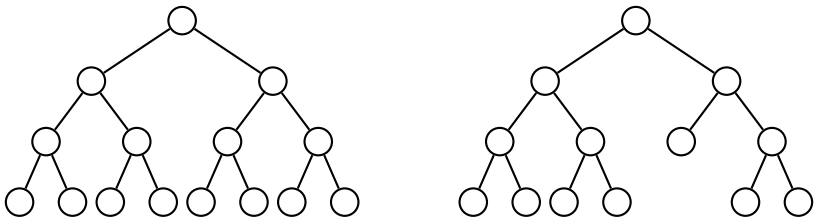
Para cada caso de prueba se escribe en una línea la lista después de eliminar los elementos.

Salida de ejemplo

```
2 5 6 9 4 10
2
2 3
4 3 5
1
```

Ejercicio 2. Árboles binarios (2,5 puntos)

Un árbol binario de altura h es *completo* cuando todos sus nodos internos tienen dos hijos no vacíos y todas sus hojas están en el nivel h. Por ejemplo, de los siguientes árboles, el de la izquierda es un árbol binario completo de altura 4, mientras que el de la derecha no es completo.



Dado un árbol binario queremos averiguar si es completo o no.
Utiliza el fichero `material/Ej2/mainEjer2.cpp`.

Entrada

La entrada comienza indicando el número de casos de prueba que vendrán a continuación. Cada caso consiste en una cadena de caracteres con la descripción de un árbol binario (correspondiente al recorrido en preorden): el árbol vacío se representa con un punto (`.`); un árbol no vacío se representa con un `*` (que denota la raíz), seguido primero de la descripción del hijo izquierdo y después de la descripción del hijo derecho.

Entrada de ejemplo

```
6
.
*..
**..*..
*.**..*..
****..*..**..*..***..*..**..*..
****..*..**..*..**.*..**..*..
```

Salida

Para cada caso, se escribirá una línea con la palabra `SI` si el árbol correspondiente es completo y la palabra `NO` en caso contrario.

Salida de ejemplo

```
SI
SI
SI
NO
SI
NO
```

Ejercicio 3. Aplicaciones de tipos abstractos de datos (5 puntos)

Estoy trabajando para una gran compañía que está desarrollando una aplicación de mensajería para teléfonos inteligentes. Esta aplicación, llamada *Mis Mensajes*, envía y recibe mensajes por internet. Además, también dispone de un complemento pensado para que los usuarios muestren su estado de ánimo a través de fotos.

Los usuarios de *Mis Mensajes* están identificados unívocamente por su número de teléfono. La información asociada a cada número de teléfono es: el nombre del usuario, la foto de perfil y las fotos del estado. Para este ejercicio tratamos todas las fotos como una cadena de caracteres sin espacios, por ejemplo, miFoto.png.

Cuando un usuario que tengo registrado pone fotos en su estado, este usuario aparece en la lista de *recientes*. Cuando veo sus fotos, el usuario deja de estar en la lista de *recientes* y se incorpora a la lista de *vistos*.

Diseña e implementa el TAD *estados* en el fichero `material/Ej3/estados.h`. Para probarlo dispones del fichero `material/Ej3/mainEjer3.cpp` que no hace falta modificar. Los errores se tratan lanzando una excepción `invalid_argument` con el nombre de la función como argumento. El TAD *estados* debe contener al menos las siguientes funciones:

- annadeUsuario(): Añade un usuario a la aplicación. Si el usuario ya existe se debe tratar el error. El usuario está determinado por su número de teléfono (string). Junto al usuario se guarda el nombre o nickname (string), la foto de perfil (string) y las fotos del estado (lista de string). Si no hay fotos de estado entonces la lista está vacía. Sin embargo, cuando añado un nuevo usuario y tiene fotos en su estado, también se inserta este nuevo usuario al inicio de la lista de *recientes*.

- annadirFotoAlEstado(): Añade una nueva foto al final de la lista de fotos del estado de un usuario dado. Si el usuario no existe se debe tratar el error. Si la lista de fotos del estado estaba vacía, al añadir una foto, este usuario se inserta al inicio de la lista de *recientes*.

- verEstado(): Imprime por pantalla todas las fotos de un usuario dado. Si el usuario no existe o no se encuentra en las lista de recientes, tratar el error. Una vez vistas todas las fotos del estado, dicho usuario deja de estar en la lista de *recientes* y se incorpora al inicio de la lista de vistos.

- verTodosLosEstados(): Imprime por pantalla todas las fotos del primer usuario que está en la lista de *recientes*, lo elimina de esta lista y lo pasa a la lista de *vistos*. Este proceso se repite hasta que la lista de *recientes* esté vacía.

- visualizaRecientes(): Visualiza todos los usuarios que están en la lista de *recientes*. Para tener más información, además de visualizar el usuario, representado por su número de teléfono, mostraremos el correspondiente nombre. El formato de salida será `usuario-nombre`.

- visualizaVistos(): Visualiza todos los usuarios que están en la lista de vistos. El formato de salida es semejante a visualizaRecientes().

Debes elegir la representación del TAD teniendo en cuenta que debes conseguir que la implementación de las operaciones sea lo más eficiente posible. Debes, así mismo, indicar justificadamente la complejidad de cada una de estas operaciones.

Entrada

La entrada está formada por una serie de casos. Cada caso termina con la palabra `FIN` y tiene una serie de líneas que muestran las operaciones a procesar seguido de los datos necesarios.

- `annadeUsuario` va seguido del número de teléfono, nombre del usuario y foto de perfil. En la siguiente línea se muestra el número de fotos de la lista fotos del estado que tiene este usuario. Si es 0 significa que no tiene fotos en su estado. Si es un número n mayor que 0 en las siguientes n líneas aparecen las fotos del estado.

- `visualizaRecientes`, `visualizaVistos` y `verTodosLosEstados` no tienen argumentos.
- `annadirFotoAlEstado` va seguido del número de teléfono y la foto que voy a añadir al final de la lista de fotos del estado.
- `verEstado` va seguido del número de teléfono del usuario del cual quiero ver las fotos de su estado.

Entrada de ejemplo

```
annadeUsuario 123456789 Juan FotoJuan.png
3
FotoJuan1.png
FotoJuan2.png
FotoJuan3.png
annadeUsuario 987654321 Juana FotoJuana.png
2
FotoJuana1.png
FotoJuana2.png
annadeUsuario 111111111 Pepa FotoPepa.png
0
visualizaRecientes
visualizaVistos
annadirFotoAlEstado 123456789 FotoJuan4.png
annadirFotoAlEstado 111111111 FotoPepa1.png
visualizaRecientes
visualizaVistos
verEstado 123456789
verEstado 987654321
visualizaRecientes
visualizaVistos
annadeUsuario 222222222 Pablo FotoPablo.png
2
FotoPablo1.png
FotoPablo2.png
verTodosLosEstados
visualizaRecientes
visualizaVistos
FIN
annadeUsuario 111 Pepe FotoPepe.png
1
FotoPepe1.png
annadeUsuario 222 Pepa FotoPepa.png
1
FotoPepa1.png
annadeUsuario 333 Ana FotoAna.png
1
FotoAna1.png
visualizaRecientes
visualizaVistos
verTodosLosEstados
visualizaRecientes
visualizaVistos
FIN
```

Salida

Para cada acción se escribirán los datos que se piden siguiendo el formato dado en el fichero `material/Ej3/mainEjer3.cpp`.

Después de cada caso se escribirá una línea con tres guiones (`---`).

Salida de ejemplo

```
Lista de recientes: 987654321-Juana 123456789-Juan
Lista de vistos:
Lista de recientes: 111111111-Pepa 987654321-Juana 123456789-Juan
Lista de vistos:
Las fotos de 123456789 son: FotoJuan1.png FotoJuan2.png FotoJuan3.png FotoJuan4.png
Las fotos de 987654321 son: FotoJuana1.png FotoJuana2.png
Lista de recientes: 111111111-Pepa
Lista de vistos: 987654321-Juana 123456789-Juan
Todos los estados: FotoPablo1.png FotoPablo2.png FotoPepa1.png
Lista de recientes:
Lista de vistos: 111111111-Pepa 222222222-Pablo 987654321-Juana 123456789-Juan
---
Lista de recientes: 333-Ana 222-Pepa 111-Pepe
Lista de vistos:
Todos los estados: FotoAna1.png FotoPepa1.png FotoPepe1.png
Lista de recientes:
Lista de vistos: 111-Pepe 222-Pepa 333-Ana
---
```